

Code responsibly with generative AI in C++ (CRWGAIC++)

ID CRWGAIC++ Preis CHF 2'250.- (exkl. MwSt.) Dauer 3 Tage

Zielgruppe

C/C++-Entwickler, die Copilot oder andere GenAI-Tools verwenden

Voraussetzungen

Allgemeine C++- und C-Entwicklung

Kursziele

- Das Wesentliche der verantwortungsvollen KI verstehen
- Vertrautmachen mit grundlegenden Konzepten der Cybersicherheit
- Korrekte Implementierung verschiedener Sicherheitsmerkmale
- Ermittlung von Schwachstellen und deren Folgen
- Lernen Sie die besten Sicherheitspraktiken in C++
- Verwaltung von Schwachstellen in Komponenten von Drittanbietern
- Ansätze und Grundsätze der Eingabevalidierung
- All dies im Kontext von GitHub Copilot

Einpacken

- Grundsätze der sicheren Kodierung
 - Grundsätze der robusten Programmierung von Matt Bishop
 - Sichere Gestaltungsprinzipien von Saltzer und Schroeder
- Und was nun?
- Quellen zur Softwaresicherheit und weiterführende Literatur
 - C- und C++-Ressourcen
 - Verantwortungsvolle KI-Prinzipien in der Softwareentwicklung
 - Generative AI - Ressourcen und zusätzliche Anleitungen

Kursinhalt

Tag 1

Verantwortungsbewusst kodieren mit GenAI

- Was ist verantwortungsvolle KI?
- Was ist Sicherheit?
- Bedrohung und Risiko
- Arten von Cybersicherheitsbedrohungen - die CIA-Triade
- Arten von Cybersicherheitsbedrohungen - das STRIDE-Modell
- Folgen von unsicherer Software
- Sicherheit und verantwortungsvolle KI in der Softwareentwicklung
- GenAI-Werkzeuge für die Kodierung: Copilot, Codeium und andere

Schwachstellen in der Speicherverwaltung

- Montagegrundlagen und Aufrufkonventionen
 - x64 assembly essentials
 - Register und Adressierung
 - Häufigste Anweisungen
 - Aufruf von Konventionen auf x64
 - Einberufung von Kongressen - was es damit auf sich hat
 - Aufrufkonvention auf x64
- Der Stapelrahmen
- Gestapelte Funktionsaufrufe
- Pufferüberlauf
 - Speicherverwaltung und Sicherheit
 - Puffersicherheitsprobleme
 - Pufferüberlauf auf dem Stack
 - Pufferüberlauf auf dem Stack - Stack Smashing
 - Ausbeutung - Entführung des Kontrollflusses
 - Übung - Pufferüberlauf 101, Wiederverwendung von Code
 - Ausnutzung - Willkürliche Codeausführung
 - Einschleusen von Shellcode
 - Übung - Code-Injektion, Ausbeutung mit Shellcode
 - Fallstudie - Stack BOF in FriendlyName Handhabung des Wemo Smart Plug
- Zeiger-Manipulation
 - Modifikation von Sprungtabellen
 - Überschreiben von Funktionszeigern
 - Bewährte Verfahren und einige typische Fehler
- Unsichere Funktionen
 - Umgang mit unsicheren Funktionen
 - Übung - Behebung eines Pufferüberlaufs (Erkundung mit Copilot)
- Verwendung von std::string in C++
 - Manipulation von C-ähnlichen Zeichenketten in C++

Code responsibly with generative AI in C++ (CRWGAIC++)

- Böartige Beendigung von Zeichenketten
- Labor - Verwirrung bei der Terminierung von Zeichenketten (Erkundung mit Copilot)
- Fehler bei der Berechnung der Stringlänge

Tag 2

Härtung der Speicherverwaltung

- Absicherung der Toolchain
 - Absicherung der Toolchain in C++
 - Verwendung von FORTIFY_SOURCE
 - Labor - Auswirkungen von FORTIFY
- Adress-Sanitizer (ASan)
 - AddressSanitizer (ASan) verwenden
 - Übung - AddressSanitizer verwenden
- Schutz vor Stapelerschlagung
 - Erkennung von BoF mit einem Stack Canary
 - Klonen von Argumenten
 - Schutz vor Stapelzerstörung auf verschiedenen Plattformen
 - SSP-Änderungen des Prologs und Epilogs
 - Labor - Auswirkungen des Stapelzerstörungsschutzes
- Laufzeit-Schutzmassnahmen
 - Laufzeit-Instrumentierung
 - Adressraum-Layout-Randomisierung (ASLR)
 - ASLR auf verschiedenen Plattformen
 - Labor - Auswirkungen von ASLR
 - Umgehung von ASLR - NOP-Schlitten
 - Umgehung von ASLR - Speicherlecks
- Nicht ausführbare Speicherbereiche
 - Das NX-Bit
 - Schreiben XOR Ausführen (W^X)
 - NX auf verschiedenen Plattformen
 - Labor - Auswirkungen von NX
 - NX-Umgehung - Angriffe durch Wiederverwendung von Code
 - Return-to-libc / Bogeninjektion
 - Rückgabeorientierte Programmierung (ROP)
 - Schutz vor ROP
- Fallstudie - Systematische Ausnutzung eines MediaTek-Pufferüberlaufs

Tag 3

Häufige Sicherheitslücken in Software

- Sicherheitsmerkmale
 - Authentifizierung
 - Passwortverwaltung
 - Verwaltung eingehender Passwörter
 - Speichern von Kontopasswörtern
 - Passwort im Transit
 - Labor - Reicht das Hashing von

Passwörtern aus?

- Wörterbuchangriffe und Brute-Forcing
- Salzen
- Adaptive Hash-Funktionen für die Passwortspeicherung
- Passwort-Politik
- NIST-Authentifikator-Anforderungen für gespeicherte Geheimnisse
- Migration der Passwort-Datenbank

- Code quality
 - Codequalität und Sicherheit
- Umgang mit Daten
 - Typ-Fehlanpassung
 - Labor - Typeninkongruenz (Erkundung mit Copilot)
 - Initialisierung und Bereinigung
 - Konstrukteure und Destruktoren
 - Initialisierung von statischen Objekten
 - Übung - Initialisierungszyklen (Erkundung mit Copilot)
 - Unveröffentlichte Ressource
 - Array-Verfügung in C++
 - Übung - Mischen von delete und delete[] (Erkundung mit Copilot)
- Fallstricke der objektorientierten Programmierung
 - Zugänglichkeitsmodifikatoren
 - Sind Zugänglichkeitsmodifikatoren ein Sicherheitsmerkmal?
 - Vererbung und Objektaufteilung
 - Implementierung des Kopieroperators
 - Der Kopieroperator und die Veränderbarkeit
 - Veränderlichkeit
 - Objekte mit veränderlicher Prädikatsfunktion
 - Lab - Objekt mit veränderlicher Prädikatsfunktion

Verwendung anfälliger Komponenten

- Sicherheit von KI-generiertem Code
- Praktische Angriffe auf Tools zur Codegenerierung
- Abhängigkeits-Halluzination durch generative KI
- Fallstudie - Eine Geschichte der Schwächen von GitHub Copilot (bis Mitte 2024)

[/list]

Code responsibly with generative AI in C++ (CRWGAIC++)

Weltweite Trainingscenter



Fast Lane Institute for Knowledge Transfer (Switzerland) AG

Husacherstrasse 3
CH-8304 Wallisellen
Tel. +41 44 832 50 80

info@flane.ch, <https://www.flane.ch>